<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1- Binary Search Trees (30pts)</td>
<td></td>
</tr>
<tr>
<td>2- Heap and Heapsort (22pts)</td>
<td></td>
</tr>
<tr>
<td>3- Sorting (10pts)</td>
<td></td>
</tr>
<tr>
<td>4- Graphs (20pts)</td>
<td></td>
</tr>
<tr>
<td>5- Hash-Table (10pts)</td>
<td></td>
</tr>
<tr>
<td>6- Linked-List (8pts)</td>
<td></td>
</tr>
<tr>
<td>TOTAL (100pts)</td>
<td></td>
</tr>
</tbody>
</table>
1- Binary Search Tree [30pts]
We consider the following array:

\[
\begin{array}{ccccccccccc}
12 & 16 & 6 & 2 & 8 & 7 & 14 & 21 & 4 & 1 \\
\end{array}
\]

[4pts] Transform the list into a binary search tree (BST). Represent the final tree structure after inserting all the elements of array A (starting from A[0] to A[9]).

[2pts] What kind of particular tree is this?

\[\text{a CBT}\]

We consider the following method1 associated with the BST structure (we consider also that a Node includes the integer variable “key”)

```java
public int method1()
{
    if (root==null){return -1;}
    Node p1=root;
    Node p2=root;
    while (p1.right!=null){p1=p1.right;}
    while (p2.left!=null){p2=p2.left; }
    return(p1.key-p2.key);
}
```

[3pts] What does method1 do?

Return the diff between max and min

[2pts] What is the value returned if applied to the BST of array A above?

20

[2pts] Explain/Comment on its BigO running times in various situations?

\[O(\log n) \text{ if tree is balanced} \]
\[O(n) \text{ if unbalanced} \]
[2pts] Provide/write the entire class Node.

```java
class Node {
    int key;
    Node left, right;

    public Node (int key) {
        this.key = key;
        left = right = null;
    }
}
```

We consider the following `method2` associated with the BST structure.

```java
public void method2() {
    if (root == null) {return;}

    Stack mystack = new Stack (100);
    Node current = root;

    while (current != null) {
        mystack.push(current);
        current = current.left;
    }

    while (!mystack.isEmpty()) {
        current = (Node) mystack.pop();
        System.out.print(current.key + " ");

        if (current.right != null) {
            current = current.right;
            while (current != null) {
                mystack.push(current);
                current = current.left;
            }
        }
    }
}
```

[4pts] What does `method2` do?

In order Traversal using a stack.

[1pts] What is the output of `method2` if applied to the BST of array A above?

```
1 2 4 6 7 8 12 14 16 21
```
[2pts] Explain/Comment on its BigO running time?
0(n), since it needs to visit all nodes

[2pts] Write a much shorter method that can do exactly the same thing without using a stack:
```java
public void inOrder(Node current) {
    if (current != null) {
        inOrder(current.left);
        System.out.println(current.key + " ");
        inOrder(current.right);
    }
}
```
We consider the following `method3` and `recMethod3` associated with the BST structure

```java
public void method3() {
    recMethod3(root);
}

public void recMethod3(Node current) {
    Node temp = current.left;
    current.left = current.right;
    current.right = temp;

    if (current.left != null)
        recMethod3(current.left);

    if (current.right != null)
        recMethod3(current.right);
}
```

[4pts] What does `method3` do?
Reverse a binary tree, mirror of the tree, symmetric with vertical axis.

[2pts] Represent the new final tree structure after calling `method3` for the BST of array A above?
```
    +--- 12
   /   /   \\
  16  6  74
 /     /     \
21 14 8  2 1
```
2- Heap and Heapsort [22pts]
[2pts] Let us consider the set of numbers: 1, 8, 4. How many valid Heaps can you obtain (largest number has the highest priority)? Represent all possible valid configurations:

[2pts] Report the main differences between insertion/removal in priority queues if implemented either by a heap or directly by an ordered array:

<table>
<thead>
<tr>
<th></th>
<th>Insert</th>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordered Array</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Heap</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

Let us consider the following array:

A=

| 12 | 16 | 6 | 2 | 8 | 7 | 14 | 21 | 4 | 1 |

[4pts] Heapify the array A using successive insertion into the heap (one item at a time, first step is given)

12

16

16

16

16

16

16

16

16

21

21

21

21

21
[2pts] Provide the CBT representation of the final heap array above.

[5pts] We now propose to heapify the array A using trickledown in place. What is the final CBT representations of the heap array?

[5pts] Heapsort in place (the first step can start with any of the heap array above)

21 16 14 12 8 7 6 2 4 1
16 12 14 4 8 7 6 2 1 21
14 18 7 4 8 1 6 2 16 21
12 8 7 4 2 1 6 14 16 21
8 6 7 4 2 1 12 14 16 21
7 6 1 4 2 8 12 14 16 21
6 4 1 2 7 8 12 14 16 21
4 2 1 6 7 8 12 14 16 21
2 1 4 6 7 8 12 14 16 21
1 2 4 6 7 8 12 14 16 21

(0.5x10)

[1pt] What is the Big-O running time complexity of heapsort?

O(N log N)

[1pt] What is the main difference between heapsort and mergesort?

Mergesort does not work in place.
3- Sorting [10 pts]

[4 pts]- Quicksort (as seen in class, pivot at the right end). Indicate the first 2 steps of the algorithm (2 partitioning steps only) applied to the array A.

Initial array

| 12 | 16 | 6 | 2 | 8 | 7 | 14 | 21 | 4 |

pivot = 1

| 16 | 6 | 2 | 8 | 7 | 14 | 21 | 4 | |

pivot = 2

| 14 | 6 | 2 | 8 | 7 | 12 | 21 | 16 | 4 |

[3 pts]- Quicksort- Given the partition function seen in class, complete the Java code shown below for the QuickSort function (3 instructions to complete).

```java
public static void quickSort(int[] values, int left, int right) {
    if (right - left <= 0) {
        return;
    } else {
        int pivotIndex = partition(values, left, right);
        quickSort(values, left, pivotIndex - 1);
        quickSort(values, pivotIndex + 1, right);
    }
}
```

Let us suppose your prefer instead using ShellSort to sort the array A.

[2 pts] What would be the resulting array just after an increment of 3 is used?

| 12 | 16 | 6 | 6 | 8 | 7 | 14 | 21 | 4 |

[1 pt] What would be the resulting array just after a new increment of 2 is used?

| 1 | 2 | 4 | 6 | 7 | 8 | 12 | 14 | 16 | 21 |
4. Graphs [20pts]

We consider the following grid-graph, the graph in non-directed and weighted (with weights ranging from 1 to 4, represented by the thickness of the edges). The 12 Vertices are numbered from bottom to top and left to right (from A to L).

Here the list of edges with thickness 1: AB1, AD1, CE1, EF1, EHI1, GHI1, GJ1, JK1, KL1
Here the list of edges with thickness 2: BE2, HK2
Here the list of edges with thickness 3: BG3, DH3, HI3, IL3
Here the list of edges with thickness 4: DE4, FI4

[2pts] What is the total weight of the graph?

\[ 6 \times 1 + 2 \times 2 + 3 \times 4 + 4 \times 2 = 9 + 4 + 12 + 8 = 33 \]

[4pts] Write its adjacency matrix (follow the alphabetical order with vertices sorted A to L)

\[
\begin{array}{cccccccccc}
A & B & C & D & E & F & G & H & I & J & K & L \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

[2pts] Starting from A, traverse the graph using BFS, and give the order of the visit (no need to consider the weights for this question). Requirement: Follow the alphabetical order.

\[ A \rightarrow B \rightarrow D \rightarrow C \rightarrow E \rightarrow G \rightarrow H \rightarrow J \rightarrow I \rightarrow K \rightarrow L \]
[6pts] Find the Minimum Spanning Tree using the Prim's algorithm. Indicate a "step by step procedure" (Vertex visited, display of the priority Queue, dequeue action). You will start with Vertex A. Represent/Plot the final MST. Give the minimum total weight.

[Diagram of a graph with vertices A, B, C, D, E, F, G, H, I, J, K, L and edges labeled with weights.]

Weight = 1 + 1 + 2 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 14

[6pts] From the complete weighted grid-graph. Find the 'cheapest' paths from Vertex A to any other vertices using the Dijkstra's algorithm. Continue to fill up this array. What is the cheapest path (with weight) to go from vertex A to vertex L?

<table>
<thead>
<tr>
<th>From</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>A1</td>
<td></td>
<td>A1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A → L

ABEHLKL 2
5- Hash Table[10pts]
[3pts] Cite three main disadvantages of using hash-tables
  - Handling collisions (hash function); fixed-size array; No reversal possible.

[3pts] A hash table of length 10 uses open addressing with hash function hash(k)=k%10, and linear probing. Represent the hash table below (values on the right), after we insert successively:
  46, 34, 42, 23, 52, 33

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>42</td>
<td>23</td>
<td>34</td>
<td></td>
<td>52</td>
<td>46</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[4pts] Complete the method find below
```
class HashTable{
    private DataItem[] table;
    private int size;

    public HashTable(int size){
        this.size=size; table = new DataItem[size];
    }
    public int hash(int key){return key%size; }

    public void insert(DataItem item){
        int h = hash(item.getKey()); // key
        while (table[h]!=null && table[h].getKey()!=-1){ // until empty spot or -1
            h=(h+1)%size; //if occupied, increment by 1
        }
        table[h]=item; // enter item into table
    }

    public DataItem find(int key){
        int h = hash(key); // Compute hash function.
        while (table[h]!=null & & table[h].getKey()!=key){
            h=(h+1)%size;
        }
        return table[h];
    }
}
```
6. Linked-List – Problem [8pts]
We propose to use a sorted double-ended doubly linked list to store a series of integer
Each link contains a publicly visible integer called value to store its contents integer.
Example with series = 3, 7, 9, 11

The DoublyLinkedList class contains the standard methods insertFirst, insertLast,
deleteFirst, deleteLast

[4pts] Write the constructor for the DoublyLinkedList class that accepts a sorted array
“array” containing the list, and initializes the linked-list.
public DoublyLinkedList(int[] array) {
    first = null;
    for (int i = 0; i < array.length; i++)
        insertLast(array[i]);
}

[4pts] We propose to implement a method that will compute the median value and return the
result. Requirement: You will use 2 probes (one traversing from the left end, the other from
the right end)
public int median()
{
    Link p1 = first;
    Link p2 = last;
    if (p1 == null) return 0;
    while (p1 != p2 && (p1.next != p2))
    {
        p1 = p1.next;
        p2 = p2.next;
    }
    if (p1 == p2) return p1.value;
    else return (p1.value + p2.value) / 2;
}

MERRY CHRISTMAS and HAPPY HOLIDAYS